Northeastern University

Data Structure & Algorithm

INFO6205

MULTILAYER - PERCEPTRON

By,

Harshitha Krishne Gowda,

Annie Joseph,

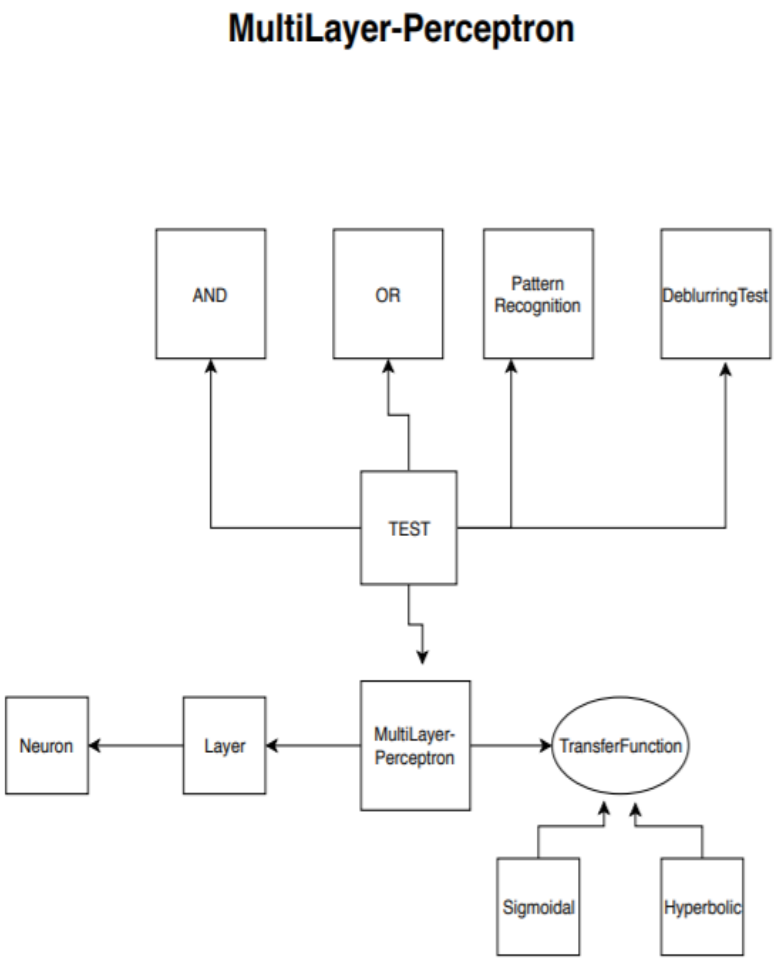
Shireen Rabbani

Problem:

To design a simple perceptron which can model an unlimited number of neurons and layers. The problem poses the situation: given a labeled dataset, it will train the model, save the model and then run the model on the test data.

Implementation Design:

The UML of the MLP model is given below:



Neuron:

This is the base class of our project and consists of the following attributes:

- Weight
- Bias
- Value
- Delta

Layer:

Since MLPs are fully connected, each node in one layer connects with a certain weight to every node in the following layer. Each layer has an array of neurons and consists of the following attributes:

- Length
- Neurons

MultiLayer – Perceptron:

This is a class of feedforward artificial neural network. It consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. It consists of the following attributes and methods:

- LearningRate
- Layers[]
- TransferFunction
- Execute()
- BackPropagate()

TransferFunction:

This is an interface which consists of the following methods:

- Evaluate
- EvaluateDerivate

SigmoidalTransfer:

This implements TransferFunction and has the following implementation for the methods:

- Evaluate: 1 / (1 + Math.pow(Math.E, - value))
- EvaluateDerivate: (value - Math.pow(value, 2))

HyperbolicTransfer:

This implements TransferFunction and has the following implementation for the methods:

- Evaluate: Math.tanh(value)
- EvaluateDerivate: 1 - Math.pow(value, 2)

Test Cases:

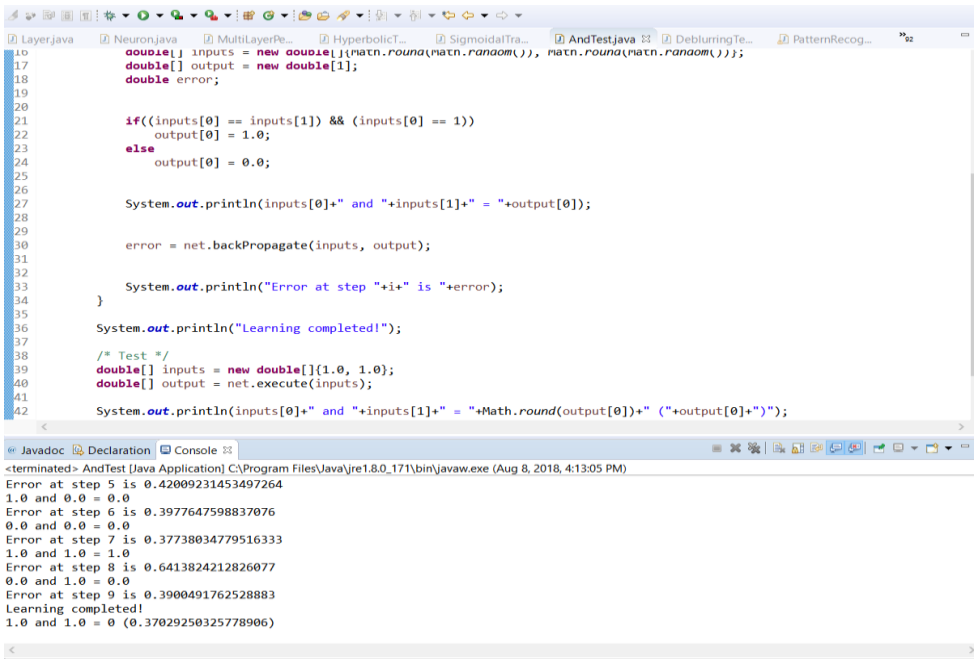
This model has been tested for the following cases:

- AND
- OR
- Pattern Recognition
- Deblurring Test

AND:

Our model successfully meets the requirements of an AND gate.
Ex:
Input = {1,1} then Output = {1}
If Input = {1,0} then Output = {0}

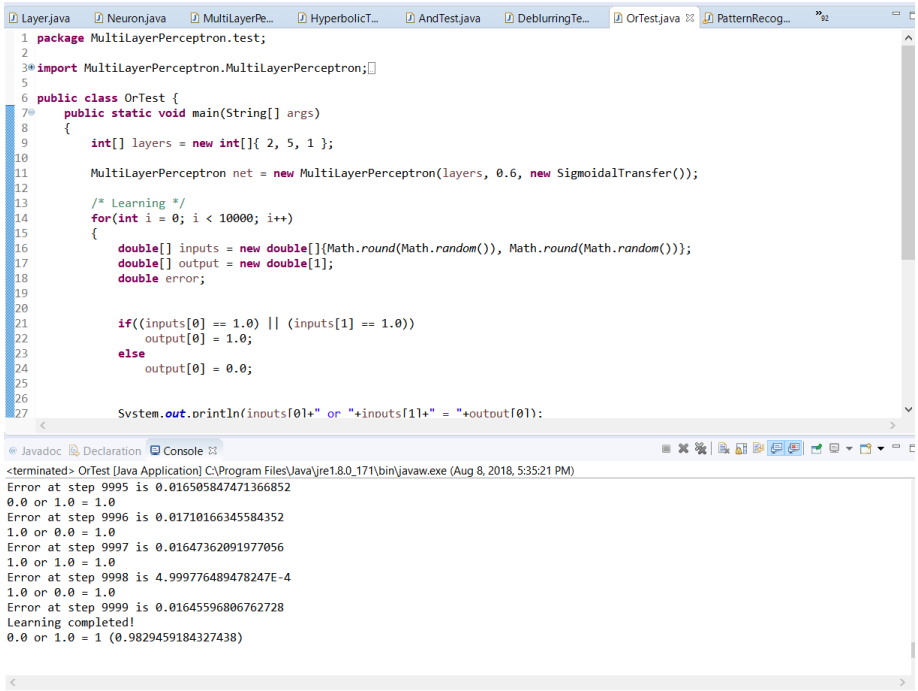
Obtained Result:



```
Error at step 5 is 0.42009231453497264
1.0 and 0.0 = 0.0
Error at step 6 is 0.3977647598837076
0.0 and 0.0 = 0.0
Error at step 7 is 0.37738034779516333
1.0 and 1.0 = 1.0
Error at step 8 is 0.6413824212826077
0.0 and 1.0 = 0.0
Error at step 9 is 0.3900491762528883
Learning completed!
1.0 and 1.0 = 0 (0.3702925032577896)
```

OR:
Our model successfully meets the requirements of an OR gate.
Ex:
Input = {1,1} then Output = {1} and if Input = {1,0} then Output = {1}
If Input = {0,0} then Output = {0}

Obtained Result:

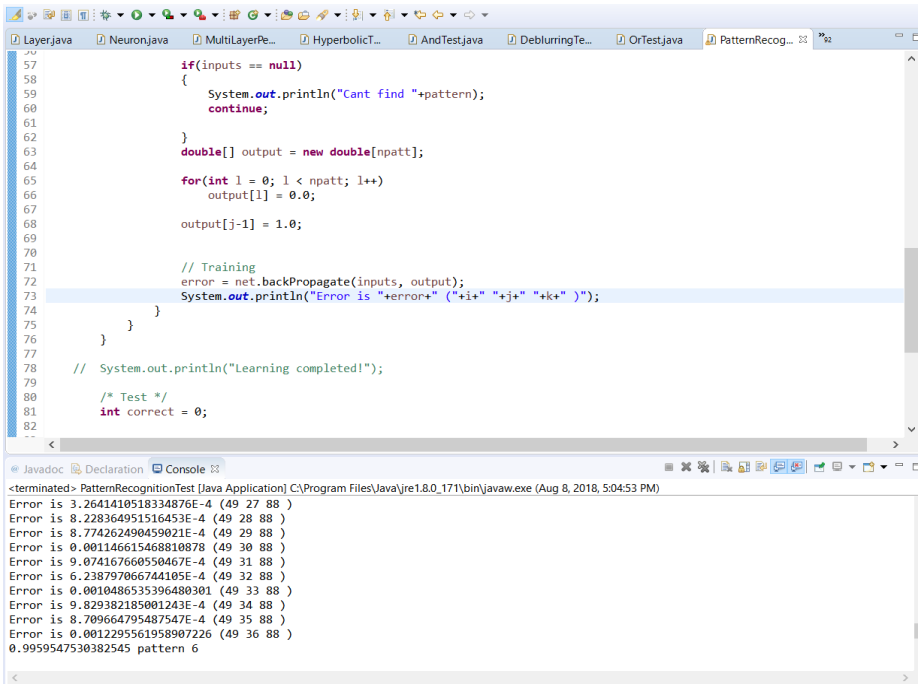


```
Error at step 9995 is 0.016505847471366852
0.0 or 1.0 = 1.0
Error at step 9996 is 0.01710166345584352
1.0 or 0.0 = 1.0
Error at step 9997 is 0.01647362091977056
1.0 or 1.0 = 1.0
Error at step 9998 is 4.999776489478247E-4
1.0 or 0.0 = 1.0
Error at step 9999 is 0.01645596806762728
Learning completed!
0.0 or 1.0 = 1 (0.9829459184327438)
```

Pattern Recognition:

Our model successfully trained and tested the neurons to produce the correct(most accurate) output for any given input.
Ex: Input = test.img(consists of a pattern to be tested) then Output = folder name containing that pattern

Obtained Result:



```
                    if(inputs == null)
                    {
                        System.out.println("Cant find "+pattern);
                        continue;

                    }
                    double[] output = new double[npatt];

                    for(int l = 0; l < npatt; l++)
                        output[l] = 0.0;

                    output[j-1] = 1.0;


                    // Training
                    error = net.backPropagate(inputs, output);
                    System.out.println("Error is "+error+" ("+i+" "+j+" "+k+" )");
                }
            }
        }

    //  System.out.println("Learning completed!");

        /* Test */
        int correct = 0;
```

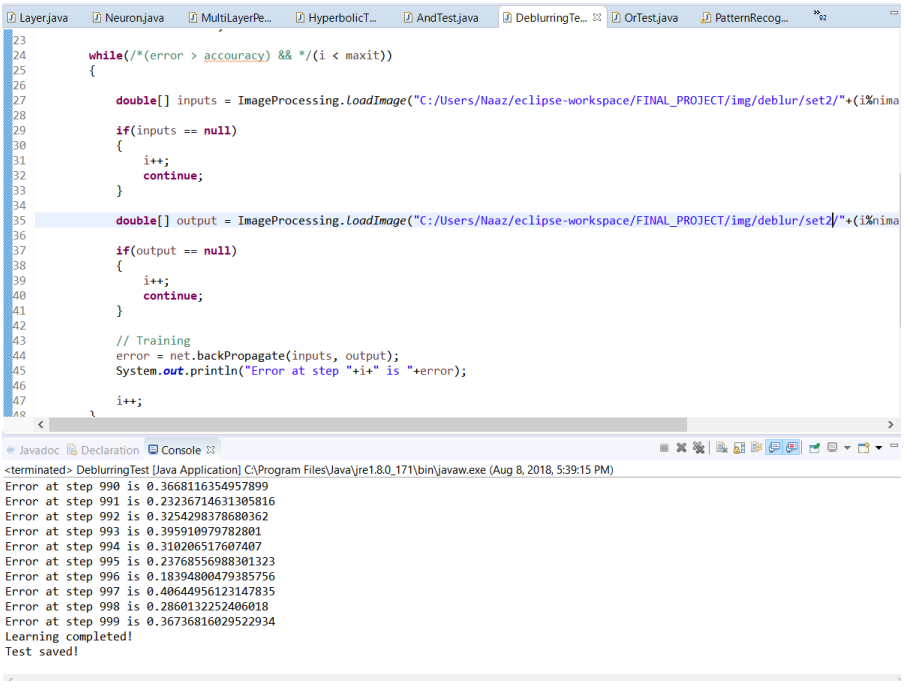
```
<terminated> PatternRecognitionTest [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Aug 8, 2018, 5:04:53 PM)
Error is 3.2641410518334876E-4 (49 27 88 )
Error is 8.2283649515164 53E-4 (49 28 88 )
Error is 8.774262490459021E-4 (49 29 88 )
Error is 0.001146615468810878 (49 30 88 )
Error is 9.074167660550467E-4 (49 31 88 )
Error is 6.2387970667441 05E-4 (49 32 88 )
Error is 0.0010486535396480301 (49 33 88 )
Error is 9.829382185001243E-4 (49 34 88 )
Error is 8.709664795487547E-4 (49 35 88 )
Error is 0.0012295561958907226 (49 36 88 )
0.9959547530382545 pattern 6
```

Deblurring Test:

Our model successfully trained and tested the neurons to produce the correct(most accurate) output for a given input.
Ex:

Input:  Output: 

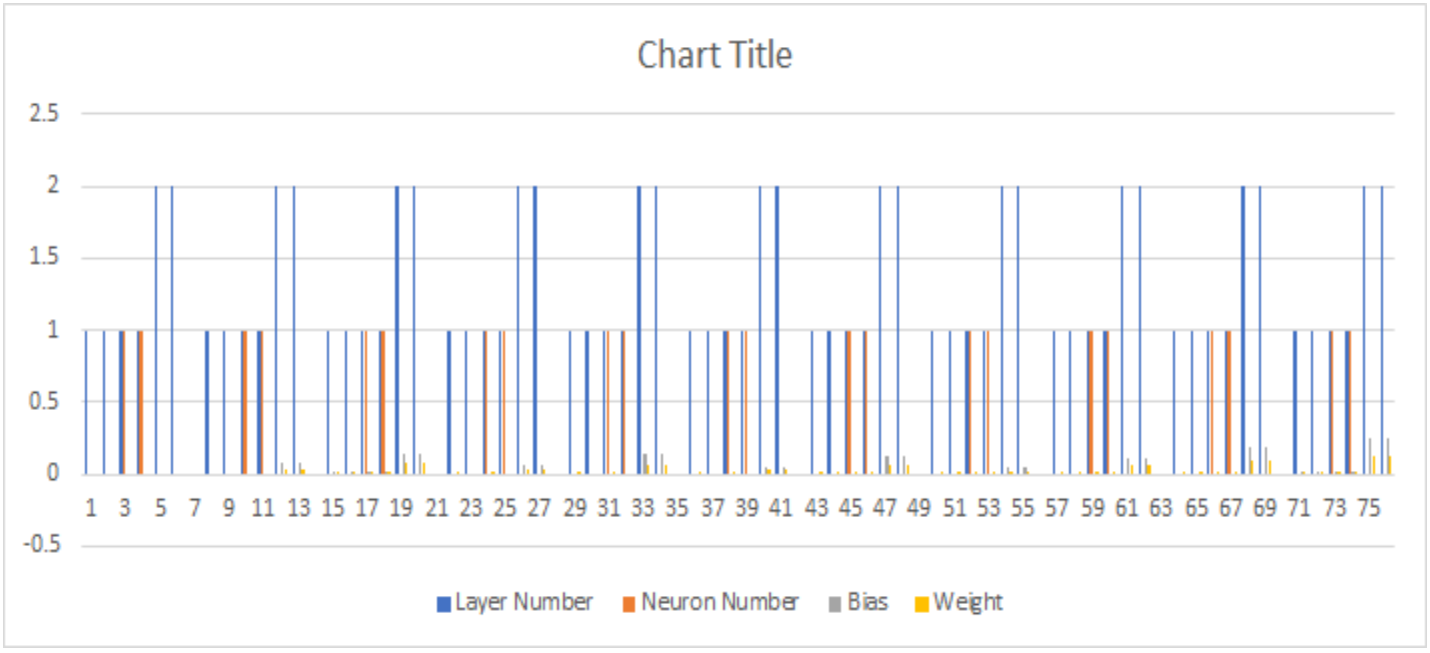
Obtained Result:



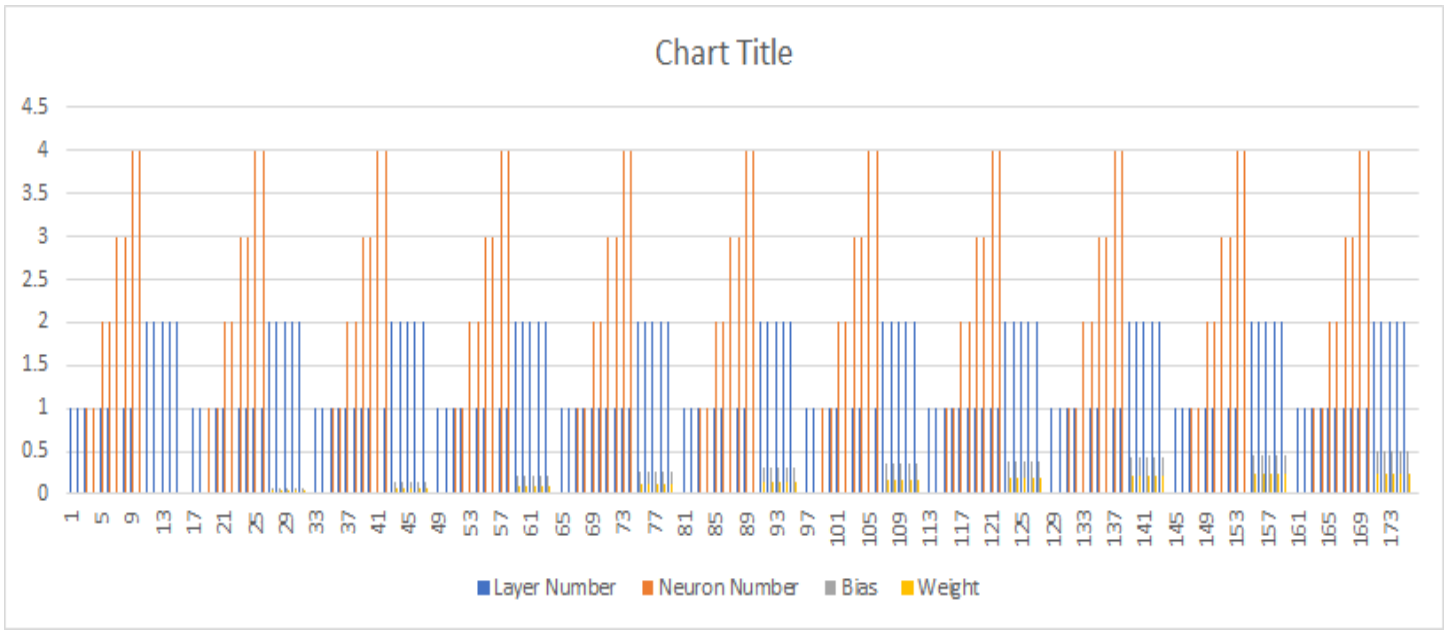
```
        while(/*(error > accuracy) && */(i < maxit))
        {

            double[] inputs = ImageProcessing.loadImage("C:/Users/Naaz/eclipse-workspace/FINAL_PROJECT/img/deblur/set2/"+(i%nima

            if(inputs == null)
            {
                i++;
                continue;
            }

            double[] output = ImageProcessing.loadImage("C:/Users/Naaz/eclipse-workspace/FINAL_PROJECT/img/deblur/set2/"+(i%nima

            if(output == null)
            {
                i++;
                continue;
            }

            // Training
            error = net.backPropagate(inputs, output);
            System.out.println("Error at step "+i+" is "+error);

            i++;
```

```
<terminated> DeblurringTest [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Aug 8, 2018, 5:39:15 PM)
Error at step 990 is 0.3668116354957899
Error at step 991 is 0.23236714631305816
Error at step 992 is 0.3254298378680362
Error at step 993 is 0.395910979782801
Error at step 994 is 0.310206517607407
Error at step 995 is 0.23768556988301323
Error at step 996 is 0.18394800479385756
Error at step 997 is 0.40644956123147835
Error at step 998 is 0.2860132252406018
Error at step 999 is 0.36736816029522934
Learning completed!
Test saved!
```

UI Report(AND Test):



UI Report(OR Test):



Confusion Matrix:

The confusion matrix obtained is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO
1	PATTERN	ACTUAL VALUE/PREDICTED VALUE	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z		TOTAL	
2	1	0	50	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		54	
3	2	1	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0		65	
4	3	2	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2		87	
5	4	3	0	0	0	100	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		109	
6	5	4	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		80	
7	6	5	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		90	
8	7	6	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		120	
9	8	7	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		85	
10	9	8	0	0	6	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		96	
11	10	9	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		70	
12	11	A	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		60	
13	12	B	0	0	0	0	0	0	0	0	0	0	0	75	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		75	
14	13	C	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		90	
15	14	D	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		80	
16	15	E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		150	
17	16	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		85	
18	17	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		95	
19	18	H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		85	
20	19	I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		100	
21	20	J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		80	
22	21	K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		70	
23	22	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		86	
24	23	M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		90	
25	24	N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		70	
26	25	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		80	
27	26	P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		60	
28	27	Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		120	
29	28	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		140	
30	29	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		70	
31	30	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		83	
32	31	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		60	
33	32	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		65	
34	33	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		90	
35	34	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		85	

Conclusion:

The above designed model successfully meets the requirements of an artificial neural network - multilayer perceptron.